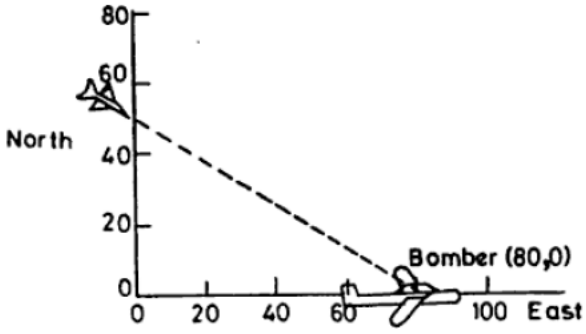
Lab sheet for Reinforcement Learning

---

1. An object moves from point A to point B step-by-step. Distance between A and B is *d*. The object moves with a constant velocity *v*. In alternating steps, the object moves forward by a random distance (between 1 and 10) and moves backward by another random distance x (between 1 and 5). How long will it take to go from point A to point B? Show the CDF for both 1D and 2D case.

   [Assume that *d*=100 meters and *v*=20 meter/s. In 2D case, the object can choose any angle randomly between 0 and 180 degrees].

2. 

   Assume that the initial positions of the fighter and Bomber are (0, 50) and (70, 0). In each step, velocities of the fighter and Bomber are changed. Velocity of fighter is random between 10 m/s and 100 m/s. On the other hand, velocity of Bomber is random between 10 m/s and 50 m/s. The fighter can hit the Bomber when the distance between them is less than 20 meters. Develop a simulator to compute the average time needed by the fighter to hit the Bomber.

3. Inventory problem:

When my stock goes down to P items (called reorder point), I will order Q more items (called reorder quantity) from the wholesaler.

If the demand on any day exceeds the amount of inventory in hand, the excess represents lost sales and loss of goodwill. On the other hand, overstocking implies increased carrying cost (i.e., cost of storage, insurance, interest, deterioration etc.). Ordering too frequently will result in excessive reorder cost.

Conditions:

(a) There is a 3-day lag between the order and arrival. The merchandise is ordered at the end of the day and is received at the beginning of the fourth day. That is, merchandise ordered on the evening of the ith day is received on the morning of the (i+3)rd day.

(b) For each unit of inventory, the carrying cost for each night is 75 USD.

(c) Each unit out of stock when ordered results in a loss of goodwill worth 2 USD per unit + loss of USD 16 net income, that would have resulted in its sale, or a total loss of USD 18 per unit. Lost sales are lost forever; they cannot be backordered.

(d) Placement of each order costs USD 75 regardless of the number of units ordered.

(e) The demand in a day can be for any number of units between 0 and 99, each equiprobable.

(f) There is never more than one replenishment order outstanding.

(g) Initially, we have 115 units on hand and no reorder outstanding.

Compare the following policies using simulation:

	P (reorder point)	*Q* (reorder quantity)
Policy I	125	150
Policy II	125	250
Policy III	150	250
Policy IV	175	250
Policy V	175	300

4. Mean response time in an M/M/1 queue:

Consider the queue shown in Figure 1. Here job sizes are i.i.d. instances of $S \sim \text{Exp}(\mu)$, where $\mu = 1$. The arrival process is a Poisson process with rate λ. The queue is called an M/M/1 to indicate that both interarrival times and job sizes are memoryless (M). For each value of $\lambda = 0.5, 0.6, 0.7, 0.8, 0.9$, record both E [T] and E [N]. Draw curves showing what happens to E [T] and E [N] as you

increase λ. To check that your simulation is correct, it helps to verify that Little's Law (E $[N] = \lambda \cdot$E $[T]$) holds. Here T represents response time of job and N represents the number of jobs in the system.

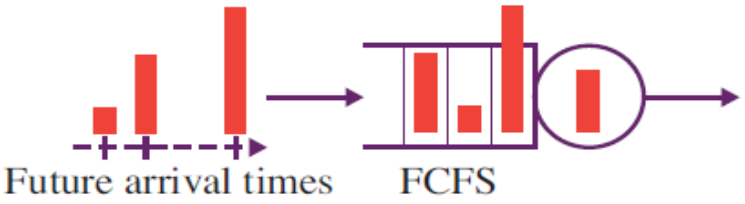


Figure 1: Queue

5. Let's see a slice of MDP:

Consider a MDP with *three* states: $\{S_0, S_1, S_2\}$, *three* actions: $\{a_0, a_1, a_2\}$, *reward function R* such that $R(S_0) = 4, R(S_1) = 7, R(S_2) = 1$, policy function $\pi(S_0) = a_1, \pi(S_1) = a_0, \pi(S_2) = a_2$;

Transition probability matrices:

Action taken: a_1	S_0	S_1	S_2
S_0	0.3	0.4	0.3
S_1	0.6	0.2	0.2
S_2	0.2	0.2	0.6

Action taken: a_2	S_0	S_1	S_2
S_0	0.5	0.4	0.1
S_1	0.3	0.2	0.5
S_2	0.4	0.2	0.4

Action taken: a_3	S_0	S_1	S_2
S_0	0.25	0.25	0.5
S_1	0.4	0.3	0.3
S_2	0.2	0.5	0.3

(a) Compute the long-term discounted reward starting from states S_0, S_1, S_2.
(b) Compute the long-term discounted reward for discount factors 0.1, 0.01, 0.001, 0.3

How long does it take to converge?

6. Estimating transition probability matrix of MDP from a given data set:

State	S_0	S_2	S_1	……..
Action taken	a_0	a_3	…	Randomly generate actions from the action space A
Next state	S_2	S_1	…	Randomly generate states from the set S

So far, we have seen MDPs where transition probability matrices are known. But in practice, they are unknown. In fact, they are computed from a given data set. In this example, first we will generate such a data set as shown in the above table. Each column of the table shows the current state, action taken and the state where the MDP has reached after taking such action. For example, initially the MDP is in state S_0. The policy has taken the action a_0 and the MDP has reached state S_2. In the next step, the policy has taken the action a_3 and the MDP has reached state S_1, and so on.

(i) Consider that the set of states S= $\{S_1, S_2, S_3, S_4, S_5\}$ and set of actions A=$\{a_1, a_2, a_3, a_4, a_5, a_6\}$. You have 1000 samples of (state, action taken, next state) triplets. Compute the following transition probability matrix from the generated data set. Each entry will look as follows: $P^{1000}_{S_5 a_1}(S_3)$, i.e., probability that the MDP will go to state S_3 if action a_1 is taken when the MDP is in S_5 as measured from 1000 samples.

(ii) Generate the transition probability matrix for 10^5 samples.

(iii) Compare the value function V (S_1) for 1000 sample and10^5 samples. What is your observation?

7. 10-armed testbed:

Consider a set of 2000 randomly generated *n-armed* bandit tasks with *n=10*. For each bandit, the true action values, q (1), q (2), …., q (10) were selected according to a *Gaussian distribution* with mean 0 and variance 1. On t^{th} time step with a given bandit, the actual reward R_t is the $q(A_t)$ for the bandit plus a normally distributed noise term having mean 0 and variance 1. Averaging over bandits, plot the performance and behavior of greedy and ε-greedy methods as they improve with experience over 1000 steps. You should plot the number of steps in the x-axis and an average reward in the y-axis. Show the impact of initialization (All zeros and All Highs).

8. Suppose we have a 20-armed bandit problem where the rewards for each of the 20 arms are deterministic and, in the range, (0, 20). Which among the following methods will allow us to accumulate maximum reward in the long term?
   (a) ε-greedy with ε= 0.1
   (b) ε-greedy with ε= 0.01
   (c) greedy with initial reward estimates set to 0
   (d) greedy with initial reward estimates set to 40

   Justify your answer through experiment.

9. Upper confidence bound action selection:

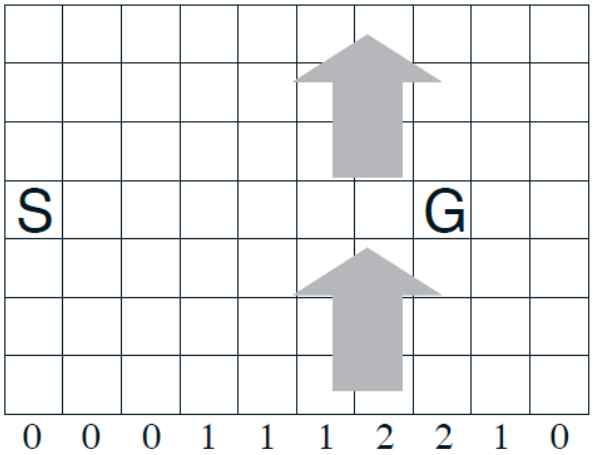
Repeat Q7 (10-armed testbed) problem with upper confidence bound action selection. Show the results for confidence level (c)=1, 2, 3 and 4 and ε=0.1, 0.01 and 0.001. Remember that the measure of uncertainty is: $c\sqrt{\frac{\ln(t)}{N_t(a)}}$. If $N_t(a) = 0$, then a is considered to be the maximizing action.

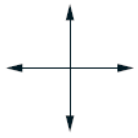
10. Temporal difference learning:

Evaluate the policies given in Q3 (Inventory problem) using Temporal difference learning (TD (0)). In this case terminal state can be defined as the state when you have *zero stock.*

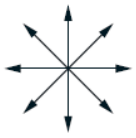
11. Windy Grid world problem (using SARSA):

Figure below shows a standard Grid world, with start and goal states, but with one deference: there is a crosswind upward through the middle of the grid. The actions are the standard four: **up, down, right, and left**; but in the middle region the resultant next states are shifted upward by a "wind", the strength of which varies from column to column. The strength of the wind is given below each column, in number of cells shifted upward. For example, if you are one cell to the right of the goal, then the action left takes you to the cell just above the goal. Let us treat this as an undiscounted episodic task, with constant rewards of -1 until the goal state is reached. Compute the time steps required to reach the goal from the initial states considering: (a) ε=0.1, α=0.5 and the initial values Q (s, a) = 0 for all s; and (b) ε=0.01, α=0.5 and the initial values Q (s, a) = 0 for all s. Show the path from source to destination.





standard moves



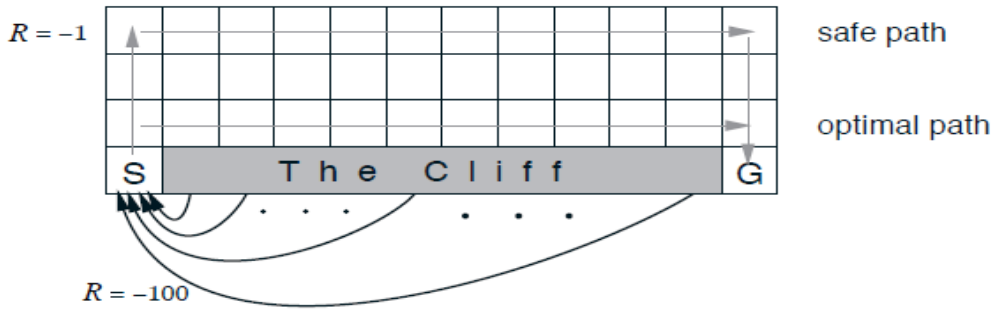
king's moves

For your reference, the SARSA algorithm is given below:

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

12. Cliff walking (Comparing Q-learning and SARSA)

Consider the Grid world shown below. This is a standard undiscounted, episodic task with START (s) and GOAL (G) states, and the usual actions causing movement up, down, right and left. Reward is -1 on all transitions except those into the region marked "The cliff". Stepping into this region incurs a reward of -100 and sends the agent instantly back to the start.



Compare the performances between SARSA and Q-learning with ε-greedy action selection for ε=0.1, 0.01 and 0.001. Show the plot on Reward per episode vs. no. of episode. For your reference, Q-learning algorithm is given below (where the symbols carry usual meaning):

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$;
  until S is terminal

←----------End of Lab sheet----------→